MIDAS: An Agent Based Data Transcoding Services Framework

Sunil Movva, Rahul Ramachandran*, Xiang Li, Sarita Khaire, Ken Keiser, Helen Conover, Sara Graves
Information Technology and Systems Center
University of Alabama in Huntsville
Huntsville, AL 35899
*rramachandran@itsc.uah.edu

***Abstract* – The objective of this agent framework is to provide end users intelligent data processing services such as subsetting and data format translation by coupling Earth Science Markup Language (ESML) interchange technology and ontologies. These ontology driven agents will be able to guide the user through the process of data processing and make decisions for them based on their output requirements. This paper describes the design approach used to extend the ESML schema to incorporate semantics using ontologies. It explains the overall architecture including the organization layers and the agent design used in this framework. A set of performatives derived from the Knowledge Query and Manipulation Language (KQML) to allow agent interaction are also be described. Finally, the paper discusses the interaction of the different components of this framework to meet user's science data preprocessing queries.**

I. INTRODUCTION

Large volumes of science data sets are being archived by different agencies in a variety of data formats. Typically, these datasets have to be preprocessed to be easily and effectively used by the scientists. *Data format translation* and *subsetting* are examples of such preprocessing steps respectively. Subsetting provides the data consumer with the capability to reduce the size and complexity of the data ordered, providing the data of interest. A vast number of applications are available which can provide data preprocessing. Most of these are either standalone or at a particular URL on the web and are often specific to a particular type of dataset. The concept of Semantic Web [1] has expanded research into the area of smart web services where the web services are described using service ontologies. Applications can look up for services on the semantic web and request these services on the fly. This provides a way to build smart and yet loosely coupled applications. However, such services have not been demonstrated in the Earth Science domain and have typically been limited to just visualization. Part of this problem is the need for a richer set syntactic and semantic metadata to allow applications to fully use the data sets. This paper describes MIDAS, an agent-based system that can provide intelligent automated data preprocessing services for Earth Science data. MIDAS provides two agent enabled data preprocessing services, Subsetting and Data Format Translation. MIDAS architecture will be discussed in this paper. Although the current prototype is not using web services, it can be easily adapted to Semantic Web and the Semantic Grid environment.

II. BACKGROUND INFORMATION

A. *Agents*

A wide range of research is going on in the area of agent technology. As the range of research has increased, a general definition of an agent has disappeared. The definition of an agent now ranges from a weak to a strict notion. For the purpose of this work, a software agent is defined as a component that adheres to the following axioms:

- Agents are autonomous.
- Agents can perceive its environment and act accordingly.
- Agents should interact with other agents.
- Agents work towards a goal.
- Agents have inferencing capability.

Characteristics of an agent are discussed in [2] and [3]. Agents can be classified into types based on their role and behavior. Agent topologies are discussed in detail in [4].

B. *Multi-Agent Systems (MAS)*

Multi-Agent Systems falls under the broad category of Distributed Artificial Intelligence (AI). A Multi-Agent System is an organization of problem-solver entities that work together to achieve a goal that is beyond the individual capabilities or knowledge of each entity [5]. MAS are generally loosely coupled. Since each agent in the system is autonomous, there is no global control of the system. Computation of any task is asynchronous which makes it suitable for the internet.

III. MIDAS SYSTEM DESCRIPTION

In order to provide intelligent data-transcoding services, an application requires a rich set of syntactic and semantic meta-data associated with the dataset. The Earth Science Mark-up Language (ESML) is a mechanism to provide full syntactic metadata for different earth science datasets [6]. In order to provide a rich set of semantic metadata for existing data sets requires creating an extension to the ESML. This extension adds a new component to the existing ESML Schema containing the semantic metadata about the dataset. The newly added component allows description of semantic metadata in a description file as instances of classes in an

external ontology. The ESML Description File now not only contains the syntactic metadata and the semantic metadata but also the link the appropriate ontology where these semantic terms are defined. An example of an ESML Description File with semantic metadata is shown in Fig.1.

A. MIDAS Design

The MIDAS design is a multi-layer model based on [7]. Each layer describes the design with respect to a particular aspect of a MAS. This ensures that all the requirements for a MAS to function are addressed. The design consists of the following layers.

- *Infrastructure Layer*: This layer provides the environment that agents can act upon, i.e. services.
- *Agent Layer*: This layer contains the design of agents used to achieve the overall goal of the framework
- *Organization Layer*: This layer defines the organizational structure of the system which is important for agents interaction
- *Coordination Layer*: This layer defines coordination methods required to resolve conflicts and select the next agent
- *Constraint Layer*: This layer verifies whether the system goals are met and interfaces with the users/user interface

Organization Layer Design: A Multi-Agent System can be viewed as an organization of groups of agents that interact to achieve a common goal. This is analogous to any corporate organizational structure, where you have a General Manager managing the organization, Divisional Managers, managing their particular divisions and employees working in particular division. Because of the scalability of this organizational hierarchy, i.e., a new division can be added as the company expands, MIDAS incorporates this organizational structure in its design.

The organizational hierarchy can be viewed as a tree structure with agents represented as nodes (see Fig.2). The root node is the Global Manager agent (analogous to a General Manager in an organization) and it controls the management domain. The next level consists of a group of Domain Manager agents (analogous to divisional managers) that report to the Global Manager. These managers are responsible for the management of service domains like a subsetting domain, data format translation domain. The lowest level consists of Worker agents associated with their corresponding services. These agents communicate with their corresponding Domain Manager Agents.

Agent Layer Design: MIDAS defines two types of agents, the Manager agents and the Worker agents. The two types differ in their role and behavior. The manager agents are concerned with allocating tasks. There are two sub-types of manager agents called the Global Manager and the Domain Manager. The Global Manager is responsible for the overall supervision of processing the user task. It breaks the task into sub-tasks and sends requests to the Domain Managers to process the sub-tasks. It keeps track of the status of the task. The Domain Manager agents are broker agents. They use advertisements to find the appropriate agent for a given task. The Worker agents in a domain advertise their capabilities with their respective Domain Manager by sending an *Advertise* message. These *Advertise* messages contain references to a service ontology (defined in DAML+OIL). The Domain Manager is integrated with a client interface to an inference engine. The specific inference engine used in MIDAS is Renamed ABox and Concept Expression Reasoner [8]. The Domain Manager uses this client interface to load the advertisements into the inference engine. As it gets a request message, it parses it to get the *Service-Concept* URI and queries the inference engine to find instances of that concept. The *Service-Concept* URI is an element of the *Evaluate* message (Fig. 3). The inference engine replies to the query with a list of Worker agent names. The Domain Manager chooses one of these agents and sends the *Evaluate* message to it. Worker agents provide one or more data transcoding services (subsetting, data translation, etc). A Worker agent translates the parameters from the users request to the parameters required by the underlying service component. The users request contains parameter values along with the URIs of parameter concepts in an ontology. The translation of parameters is done with the help of this ontology and an inference engine. The underlying service components are coupled with a Worker agent and form the lowest infrastructure layer (Fig. 2). The service component also registers the URIs of the parameters that it expects.

Communication Protocol: The critical component in a MAS is the communication protocol used between the agents. This is because agents in a MAS need to provide and request information in order to achieve their goals. Unlike a tightly coupled system that depends on set of function calls, MIDAS uses a declarative approach. Communication between agents is accomplished through a blackboard. This type of architecture is often called blackboard architecture and is a foundation for modern user interface architectures. A blackboard is a common place for agents to read and write messages. An agent wanting to send a message puts the message on the blackboard. This message is tagged with sender and receiver information. The receiving agent checks the receiver information and understands that the message is for it. A blackboard is associated with each domain and is used for communication between agents in that particular domain. Since Domain Managers co-exist in two domains, viz. Global-domain (Management Domain) and the Service-domain (Subsetting Domain), they have access to both the blackboards (see Fig 2).

A subset of a agent communication language called the Knowledge Query and Manipulation language (KQML) [9] is

adapted as a communication protocol for MIDAS. The adapted version is XML based. Each message consists of two levels of information. The communication level information contains information such as sender, receiver, message id and message type. Whereas the content information contains specific content information pertaining to the type of message. Fig.3 depicts the XML schema for the *Evaluate* message. The content information is embedded in the <*content*> element of the message. The communication information is understandable (parsable) by all the agents in MIDAS, while only the specific receiver agent parses the content information. All the agents in the system are integrated with a communication component. This component can read the standard messages and write the standard messages to the associated message board. The information inside the <*content*> element is not parsed by this component. It is extracted out and passed to the associated agent for further processing. The set of standard messages used are listed below.

Basic Message Set Used in MIDAS:

- `Advertise:` This message indicates that the sender is particularly suited to process a particular type of requests.
- `Unadvertise:` This message indicates that the sender is no longer willing to process a particular type of advertised requests.
- `Error:` This message indicates that the sender cannot understand or process a particular message.
- `Sorry:` This message indicates that the sender understands, but is not able to provide any (more) response(s) to a particular message.
- `Evaluate:` This message is a request to the recipient to process the request. An example request is 'subset' with some parameters.
- `Reply:` This message is a response to a message from the receiver. It can be a response to a 'evaluate' message.
- `Ask-Status:` This message is a request to a send the status of a job or an agent.

Coordination Layer Design: One of the main characteristics of an agent is that it is autonomous. Therefore, a Worker agent should be able to stop providing its service or a Domain Manager should be able to stop serving with out disrupting the whole system. This requirement is met by using the broker design to perform the coordination between different layers. The Domain Manager acts as a broker agent. Any worker agent going out of service sends a *unadvertise* message to the Domain Manager. This design reduces the responsibilities of the Global Manager as it need not keep track of individual Worker agents. The Global Manager simply passes on the request to the right Domain Managers. Unlike a tightly coupled matchmaking approach, this design reduces the search time required to find the right Worker agents upon receiving a request.

Constraint Layer Design: This layer design provides a way to verify the status of a goal and provide the user with the results. A simple structure called the task tree serves this purpose. A task-tree is a simple tree structure with nodes representing a task and the subnodes of a node represent the subtasks. Each node has a task status attribute. Once the task assigned to a Worker agent is completed, it reports the result back to its Domain Manager. The Domain Manager in turn sends the result with a reply message to the Global Manager, which sets the task status attribute associated with that task. The Global Manager can find out the status of a task at any moment by sending an 'ask-status' message to the Domain Managers.

B. *Working of the System*

Once a user submits a request to MIDAS, the Global Manager parses this request into subtasks and builds a task tree from it. The Global Manager multi-casts the *evaluate* requests for the subtasks to all the Domain Managers, starting from the bottom of the tree traversing inorder. This multicast is done by writing a sequence of *evaluate* messages with *receiver=all* to the global blackboard. Each Domain Manager maintains a list of Worker agents in its domain along with their advertisements. As a Worker agent comes into existence, it advertises its capabilities to the corresponding Domain Manager by sending an *advertise* message. The Domain Manager loads these advertisements into the inference engine. Upon receiving a service request, the Domain Manager sends a query to the inference engine to find instances of a requested service. After receiving the reply for a match, the Domain Manager sends an evaluate request to the particular agent by placing a *evaluate* message for this agent in the local blackboard. The corresponding worker agent reads the message from the blackboard. The *evaluate* message contains parameters required for the data preprocessing service. The Worker agent parses these parameters, maps them to the parameters required by the service API and invokes the service. The overall interaction process between the components is depicted in Fig.4. Once the task is complete, the Worker agent reports to the Domain Manager by placing a *reply* message on the local blackboard. This message contains status as well as the result of a task. The Domain Manager in turn reads this message and sends a reply to the original request from the Global Manager. The Global Manager now reads this reply and resets the status of the task in the task-tree. This process continues in parallel until all the sub-tasks in the task tree are completed. Upon completion, the Global Manager sends the result to the user-interface component. Although, the processing of a single task is sequential, the system overall is asynchronous.

IV. FUTURE WORK

Although, MIDAS currently only has two domains, subsetting and format translation, the design is scalable. It allows addition of new domains to the overall framework seamlessly. One can extend the functionality of MIDAS by adding other domains like 'data search' services where multiple Worker agents catalog and maintain separate metadata archives. Adapting MIDAS to work with distributed web services will be investigated as part of the future work. In addition, migrating MIDAS to the grid environment to address Semantic Grid notion will also be explored.

ACKNOWLEDGEMENT

The authors acknowledge support from the Earth Science Technology Office award NAG5-13575, Goddard Space Science Flight Center, NASA.

REFERENCES

[1] Berners-Lee, T., Hendler, J. and Lassila, O., "The Semantic Web", Scientific American, 284, pages 34-43 May 2001.

[2] Wooldridge, M. and Jennings, N.R., "Intelligent Agents: Theory and Practice" Knowledge Engineering Review, Oct 1994.

[3] Russel, S., and Norvig, P., "Artificial Intelligence: A Modern Approach", Second Edition, Prentice Hall, Upper Saddle River, N.J., 2003.

[4] Hyacinth, S. Nwana, "Software Agents: An Overview", Knowledge Engineering Review, Sep 1996.

[5] Durfee, E.H., Lesser, V.R. and Corkill, D.D., "Trends in Cooperative Distributed Problem Solving", IEEE Transactions on Knowledge and Data Engineering, KDE-1(1), pages 63-83, Mar 1989.

[6] Ramachandran, R., Graves, S., Conover, H., and Moe, K., "Earth Science Markup Language (ESML): a solution for scientific data-application interoperability problem", Computers & Geosciences, 30(1): 117-124, 2004.

[7] Lee, S.K. and Hwang, C.S., "Architecture layers and engineering approach for agent-based system", Information and Software Technology 45, pages 889–898, Mar 2003.

[8] Haarslev, V. and Moller, R., "RACER System Description", In Proceedings of the International Joint Conference on Automated Reasoning, IJCAR'2001.

[9] Genesereth, M. and Fikes, R., "Knowledge Interchange Format", Version 3.0 Reference Manual, Technical Report, Computer Science Department, Stanford University, USA, 1992.

```
<a:ESML xmlns:a="ESML"
        xmlns:ns0="file:/C:/Sunil/MiniMDAS/MDASOntology.daml#"
        xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="ESML
        ESML.xsd">
        <SemanticMetaData>
                <rdf:Description rdf:about=" #Test">
                <rdf:type>
                        <daml:Class rdf:about="ns0:Structure"/>
                </rdf:type>
                        <ns0:contains rdf:resource=" #Lat"/>
                        <ns0:contains rdf:resource=" #Lon"/>
                        <ns0:contains rdf:resource=" #Time"/>
                        <ns0:contains rdf:resource=" #Field1"/>
                        <ns0:contains rdf:resource=" #Field2"/>
                </rdf:Description>
                <rdf:Description rdf:about=" #Lat">
                    <rdf:type>
                      <daml:Class rdf:about="ns0:Latitude"/>
                    </rdf:type>
                    <ns0:contains rdf:resource=" #Degrees"/>
                        <ns0:contains rdf:resource=" #Scale-1"/>
                        <ns0:contains rdf:resource=" #Offset0"/>
                </rdf:Description>

                …
                <rdf:Description rdf:about="#Field1">
                    <rdf:type>
                      <daml:Class rdf:about="ns0:DataField"/>
                    </rdf:type>
                </rdf:Description>
                …
        </SemanticMetaData>
        <SyntacticMetaData>
                <Ascii>
                        <Structure name ="Test" instances="1">
                                <Array occurs="5">
                                        <Array occurs="5">
                                                <Field name="Lat" format="%d"/>
                                        </Array>
                                </Array>
                …
        </SyntacticMetaData>
</a:ESML>
```

Fig. 1. ESML Description File with the additional semantic component

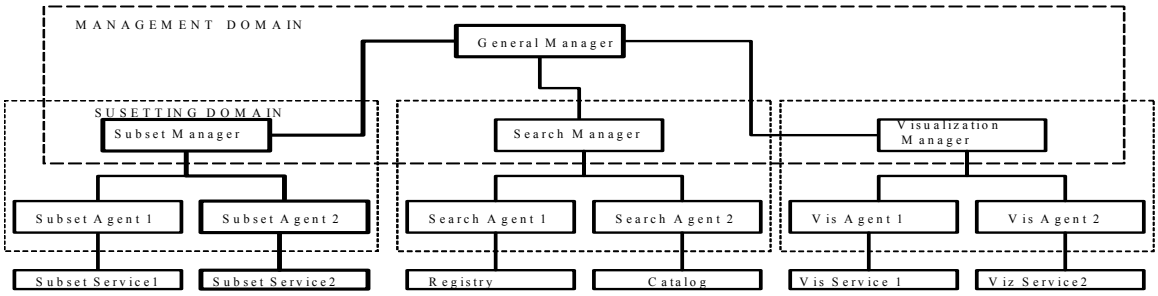

Fig. 2. MIDAS Organizational Structure

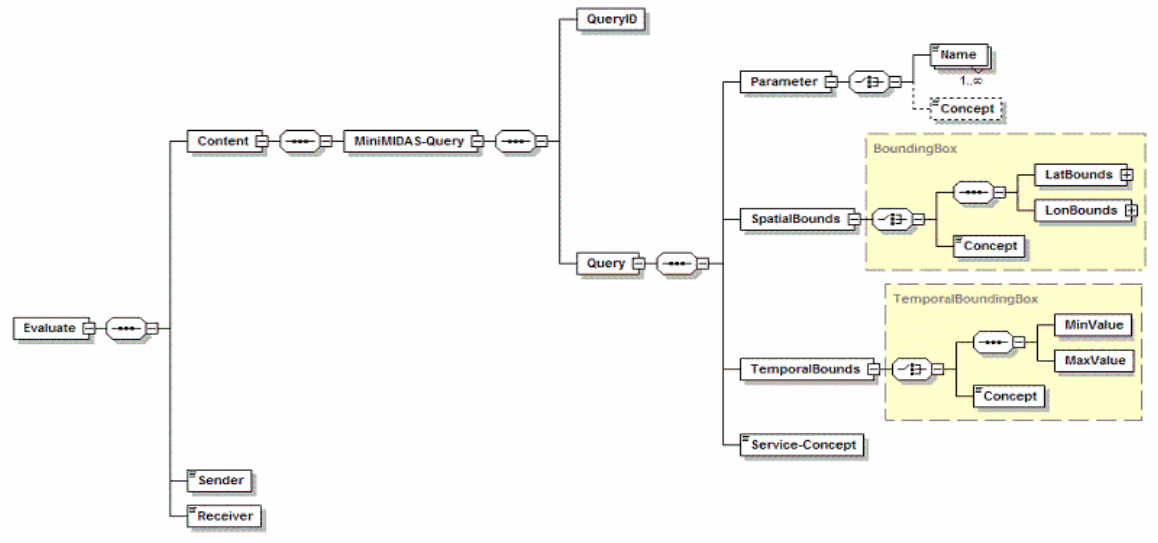

Fig. 3. Schema for the Evaluate message

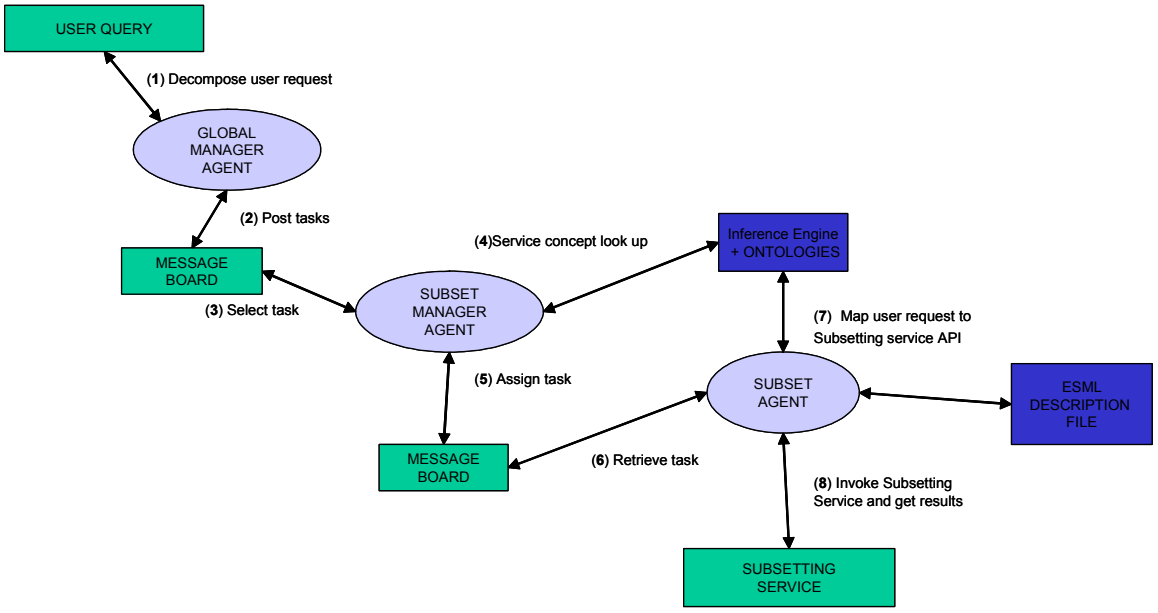

Fig. 4. Example of task processing in MIDAS